ALGORITHMS FOR BICRITERION AND CONSTRAINED SCHEDULING PROBLEMS

A Thesis Submitted
In Partial Fulfilment of the Requirements
for the Degree of
MASTER OF TECHNOLOGY

by

A. R. KRISHNA

to the
INDUSTRIAL & MANAGEMENT ENGINEERING PROGRAMME
INDIAN INSTITUTE OF TECHNOLOGY, KANPUR
JULY, 1982

MEP-1982-M-KRI-ALG

CERTIFICATE

This is to certify that the work embodied in the thesis "ALGORITHMS FOR BICRITERION AND CONSTRAINED SCHEDULING PROBLEMS", by A. Radha Krishna has been carried out under my supervision and has not been submitted elsewhere for a degree.

(J. L. Batra)
Professor and Head
Industrial and Management Engg.Prog
Indian Institute of Technology,
Kanpur 208016

July, 1982

ACKNOWLEDGEMENTS

I am grateful to Dr. J.L. Batra for his continuous encouragement and guidance throughout the span of this work. I am also grateful to Dr. R.K. Ahuja for his encouragement and timely help throughout this work

I am also thankful to other faculty members and friends for their help and encouragement during my stay in this Institute.

I am thankful to Mr. J.K. Misra for excellent typing.

A. Radha Krishna

CONTENTS

ABSTRACT

In this thesis we consider some generalisations of single criterion scheduling problems. The first problem considered is a bicriterion scheduling problem in which the two criteria are minimisation of the weighted mean flowtime with respect to two sets of weights. A simple and efficient method is suggested which enumerates a substantial number of efficient solutions of the bicriterion scheduling problem. A variant of this bicriterion scheduling problem is also considered in which one of the criteria is changed into a constraint. Finally, we consider another constrained scheduling problem, where the sum of weighted completion times is minimised subject to a due date constraint. A branch and bound algorithm and a heuristic method are proposed to solve this problem. All the suggested algorithms are coded in FORTRAN-10 and implemented on DEC-1090 time-sharing computer system. Extensive computations are performed for each algorithm for various sizes of the randomly generated problems. Results of these computations are reported which are quite encouraging. Performance of the heuristic methods is specially noteworthy which are able to solve large-sized problems with considerable accuracy in reasonable amount of time.

CHAPTER I

INTRODUCTION

1.1 Introduction:

Scheduling is the allocation of resources over time to perform a collection of tasks. The practical problem of allocating resources over time to perform a collection of tasks arises in a variety of situations. Scheduling theory is concerned primarily with mathematical models that relate to the allocation of resources over time, and the development of useful models and techniques. In particular, the quantitative approach begins with a translation of decision making goals into an explicit objective function and decision making restrictions into explicit constraints.

Scheduling problems are much prevalent in real life situations. They exist whenever there is a choice as to the order in which a number of tasks can be performed with respect to time. A problem could involve: jobs in a manufacturing plant, aircraft waiting for landing clearances, bank customers at a row of tellers' windows or programs to be run at a computing centre.

Nevertheless, three types of decision making goals seem to be prevalent in scheduling decisions: Efficient utilisation of resources, rapid responses to demands, and close conformance to prescribed deadlines. Two kinds of feasibility constraints are commonly found in scheduling problems. First, there are limits on the capacity of available resources and, second, there are technological restrictions on the order in which tasks can be performed. A specific scheduling problem is described by four types of information:

(i) The jobs and operations to be performed,
(ii) The number and types of machines that comprise the shop,
(iii) Desciplines that restrict the manner in which assignments can be made,
(iv) The criteria by which a schedule will be evaluated.

Most of the scheduling problems considered so far, in literature, are single criterion scheduling problems. The most widely used criteria are minimisation of

(i) makespan time,
(ii) weighted mean flowtime,
(iii) number of tardy jobs in the system,
(iv) maximum tardiness.

The problems considered in this thesis are generalisations of the single criterion scheduling problems. First, we

consider a n job, one machine bicriterion scheduling problem. The criteria are to minimise the weighted mean flowtime with respect to two sets of weights. We propose a heuristic method to obtain a substantial number of efficient solutions. We also consider a variant of the above problem in which one of the criteria is changed into a constraint. Finally, we consider a constrained scheduling problem in which we minimise the sum of weighted completion times subject to a due date constraint. Heuristic methods are proposed for these scheduling problems. Branch and bound algorithms are also suggested to test the effectiveness of the heuristic methods.

1.2 Preliminaries:

1.2.1 Bicriterion Scheduling Problem:

In most of the scheduling problems, multiplicity of criteria for judging the alternatives is pervasive. That is, for many such problems, the decision maker wants to attain more than one objective goal in selecting the course of action while satisfying the constraints dictated by the environment, processes, and resources. Another characteristic of these problems is that the objectives are apparantly non-commensurable i.e. it is not possible to combine any two criteria into one criterion by attaching suitable weights to different criteria. When we have two criteria to be considered, we call such problems as bicriterion scheduling problems. Mathematically, these problems can be represented as:

$$\text{Min} \quad (f_1(X),\ f_2(X))$$

$$\text{S.t.} \quad g_i(X) \geq 0, \qquad i = 1, \ldots, m$$

where X is an n dimensional decision variable vector.

One approach for solving this type of problems is to enumerate the efficient solutions, which are defined below, and pick-up one after subjective judgement. The other approach is, to optimise one of the objectives while appending the other objective to the constraint set so that optimal solution would satisfy the other objective atleast upto a predetermined level. The problem is given as:

$$\text{Min.} \quad f_2(X),$$

$$\text{S.t.} \quad g_i(X) \geq 0, \qquad i = 1, \ldots, m$$

$$f_1(X) \leq F_1,$$

where F_1 is the acceptable predetermined level for objective 1.

Efficient Solution:

An efficient solution is one in which one objective cannot be improved without a simultaneous increment to the other objective. That is, X^* is an efficient solution to the bicriterion scheduling problem if there does not exist any $\overline{X} \in X$, such that

$$f_1(\overline{X}) \leq f_1(X^*) \quad \text{and} \quad f_2(\overline{X}) \leq f_2(X^*)$$

with atleast one inequality.

1.2.2 Heuristic Methods:

The heuristic method can be defined as solving problems by an intuitive approach in which the structure of the problem can be interpreted and exploited intelligently to obtain a near optimum solution in reasonable amount of time. There are several possible reasons for using a heuristic method: (i) The mathematical problem is of such a nature that an analytic or iterative solution procedure is unknown, (ii) Although an analytic or iterative solution procedure may exist, it may be computationally prohibitive or perhaps unrealistic in its data requirements, (iii) The heuristic method by design may be simpler for the decision maker to understand, hence markedly increasing the chances of implementation.

A good heuristic method should possess the following properties (i) realistic computational effort to obtain the solution, (ii) the solution should be close to the optimum on the average, (iii) the chances of a very poor solution should be low, (iii) the heuristic should be as simple as possible for the user to understand, preferably explainable in intuitive terms, particularly if it is to be used manually.

1.2.3 Notations and Definitions:

We now present some notations, definitions and theorems which are used in this thesis.

1. Permutation schedule (π): These are the schedules that are completely specified by giving the order in which jobs will be processed. Such an ordering is described by giving one of the n possible permutations of the job identification numbers 1,2,..., n. Given the specification of the sequence and the job processing times, one can calculate the values of completion times and from them any property of the schedule.

2. Processing time (p_j): The amount of time required for processing the job j.

3. Ready time (r_j): The time at which job j is available for processing.

4. Completion time (c_j): The time at which the processing of job j is completed.

5. Flowtime (F_j): The amount of time job j spends in the system. It is obvious that $F_j = c_j - r_j$.

6. Due date (d_j): The time at which processing of job j is to be completed.

7. Lateness (L_j): The amount of time by which the completion time of job j exceeds the due date.

8. Tardiness (T_j): The lateness of the job j if it fails to meet the due date. It is obvious that $T_j = \max(0, L_j)$.

9. Weight (W_j): The importance assigned to job j.

10. Mean flowtime ($\bar{F}$): If n jobs are to be scheduled, mean flowtime is,

$$\bar{F} = \frac{1}{n} \sum_{j=1}^{n} F_j.$$

11. Weighted mean flowtime ($\bar{F}_w$): If n jobs are to be scheduled, weighted mean flowtime is

$$\bar{F}_w = \sum_{j=1}^{n} w_j F_j / \sum_{j=1}^{n} w_j .$$

12. Maximum tardiness (T_{max}): If n jobs are to be scheduled, maximum tardiness is,

$$T_{max} = \max_{1 \leq j \leq n} \{T_j\}.$$

13. $\pi(j)$: is the job in jth position in the sequence π.

1.2.4 Theorems:

Some of the known results used in this thesis are given below.

Theorem 1: Mean flowtime ($\bar{F}$) is minimised by the following sequence π, known as the shortest processing time (SPT) sequence:

$$p_{\pi(1)} \leq p_{\pi(2)} \leq \cdots \leq p_{\pi(n)}.$$

Theorem 2: Weighted mean flowtime ($\bar{F}_w$) is minimised by the sequence π, known as the Weighted Shortest Processing Time (WSPT) sequence:

$$\frac{w_{\pi(1)}}{p_{\pi(1)}} \geq \frac{w_{\pi(2)}}{p_{\pi(2)}} \geq \cdots \geq \frac{w_{\pi(n)}}{p_{\pi(n)}} .$$

Theorem 3: The maximum tardiness (T_{max}) is minimised by the following sequence π known as Earliest Due Date (EDD) sequence:

$$d_{\pi(1)} \leq d_{\pi(2)} \leq \cdots \leq d_{\pi(n)}.$$

1.3 Outline of the Thesis:

A brief, chapter by chapter, outline of the thesis is given in this section.

In Chapter II, we consider a bicriterion scheduling problem. The problem is to sequence n jobs on a single machine to minimise the weighted mean flowtime with respect to two sets of weights. The problem is as follows:

$$\underset{\pi \in S}{\text{Min}} \left\{ \begin{array}{l} \dfrac{1}{\sum_{j=1}^{n} w^1_{\pi(j)}} \sum_{j=1}^{n} w^1_{\pi(j)} F_{\pi(j)} \\ \dfrac{1}{\sum_{j=1}^{n} w^2_{\pi(j)}} \sum_{j=1}^{n} w^2_{\pi(j)} F_{\pi(j)} \end{array} \right\}$$

where S is a set of permutation schedules, π is a permutation schedule, $F_{\pi(j)}$ is the flowtime of job in the jth position of sequence π, and $w^1_{\pi(j)}$, $w^2_{\pi(j)}$ are the two sets of weights. We propose a heuristic method to enumerate a substantial number of efficient solutions. The heuristic is computationally tested for its accuracy, and its effectiveness to solve large sized problems.

In Chapter III, we consider a constrained scheduling problem. This problem is to sequence n jobs on a single machine to minimise the sum of weighted completion times subject to the constraint that the maximum tardiness does not

exceed the maximum tardiness obtained from the EDD sequence. The problem is as follows:

$$\underset{\pi \in S}{\text{Min}} \quad \sum_{j=1}^{n} w_{\pi(j)} \, c_{\pi(j)},$$

$$\text{S.t.} \quad c_{\pi(j)} - d_{\pi(j)} \leq T,$$

$$\text{where } c_{\pi(j)} = \sum_{i=1}^{j} p_{\pi(i)},$$

where $c_{\pi(j)}$ is the completion time of the job in jth position in the sequence π, and $d_{\pi(j)}$ its due date. Burns [3] has proposed an algorithm for the above given problem and claimed that it yields a local optimum solution. In this chapter, we show with the aid of a numerical example that Burns claim is incorrect. We propose a heuristic method which gives a locally optimum solutions. A branch and bound method is also suggested to obtain a globally optimum solution. These are coded and tested computationally and results are reported.

In Chapter IV, we study a modified version of the scheduling problem presented in Chapter II. One of the criteria is changed into a constraint. The problem is as follows:

$$\underset{\pi \in S}{\text{Min}} \quad \frac{1}{\sum_{j=1}^{n} w_{\pi(j)}^{2}} \sum_{j=1}^{n} w_{\pi(j)}^{2} \, F_{\pi(j)}$$

$$\text{S.t.} \quad \frac{1}{\sum_{j=1}^{n} w_{\pi(j)}^{1}} \sum_{j=1}^{n} w_{\pi(j)}^{1} \, F_{\pi(j)} \leq F_1 ,$$

where F_1 is an aspiration level on the first criterion. We propose a heuristic method for solving this problem. A branch and bound algorithm is also proposed for this problem. Both of them are computationally tested and the results are reported.

Since the problems considered in various chapters are different, literature is surveyed chapterwise.

CHAPTER II

A BICRITERION SCHEDULING PROBLEM

2.1 Introduction:

Most of the industrial scheduling problems are essentially multiple criteria optimisation problems. The makespan time, the weighted mean flow time and tardiness etc. are some of the criteria which must be considered before arriving at the final schedule. Panwalkar's study [8], based on several real life scheduling problems, also found this as its major conclusion. Nevertheless, scheduling research continued to deal predominantly with single criterion. It is clear, however, that a socalled optimum with respect to one criterion could perform extremely bad with respect to the other criteria. Therefore, a non-optimal solution with satisfactory performance on other measures might be considered as a better alternative by the decision maker. This point was partly recognised by some researchers who studied scheduling problems with secondary criteria (See [2,3,4]). These studies identify the best sequence among the set of alternate optima with respect to the primary measure.

Not much work has been reported in the area of bicriterion scheduling. Emmons [4], considered one machine sequencing problem to minimise mean flow time with minimum number of tardy jobs. He introduced a branch and bound algorithm to produce the optimum schedule. He also proposed a heuristic method to obtain a good, and often optimal, schedules. Vanwessenhove [10], has considered one machine sequencing problem with minimum mean flow time and minimum maximum tardiness as two criteria. He suggested a pseudo-polynomial algorithm to enumerate all the efficient solutions.

In this chapter, another bicriterion scheduling problem is considered in which the two criteria are minimisation of the weighted mean flow time with respect to two sets of weights. For example, let us consider n machines to be repaired by a single operator. He wants to minimise the weighted mean down time of the machines. He may also be interested in minimising the average number of jobs in the system. This can be formulated as a bicriterion scheduling problem with two sets of weights.

In this chapter, a method is developed to enumerate a substantial number of efficient solutions of the problem considered. The initial efficient solution is obtained by minimising the first criterion only. The other efficient solutions are obtained by adjacent pairwise exchanges leading to maximum decrease in the value of second objective function by incurring

minimum increase in the value of first objective function. The method terminates when a sequence optimum with respect to the second objective function is obtained. Computational experience shows that 90 percent of the time, this method enumerates atleast 50 percent of the efficient solutions and on an average it enumerates 70 percent of the efficient solutions. The other attractive features of the heuristic method are its simplicity, intuitive appeal and its polynomial boundedness.

The brief summary of this chapter is as follows: The problem statement and its solution methodology are presented in Secs.2.2 and 2.3 respectively. A step by step illustration of the solution methodology for a selected problem is given in Section 2.4. Computational results are reported in Sec. 2.5.

2.2 The Problem Statement:

Consider the problem of sequencing n jobs on a single machine with ready times as zero, i.e., all the jobs are available at the same time. Let $p_{\pi(j)}$ denote the processing time of job in the jth position of sequence π, and $w^1_{\pi(j)}$, $w^2_{\pi(j)}$ be the two weights associated with it. The two criteria considered are (i) minimisation of weighted mean flow time with respect to $w^1_{\pi(j)}$ as a set of weights, and (ii) minimisation of weighted mean flow time with respect to $w^2_{\pi(j)}$ as another set of weights.

The following notations are used:

S : the set of permutation schedules,

π : a permutation schedule,

$F_{\pi(j)}$: the flow timeof job in the jth position of sequence π.

The Bicriterion Scheduling Problem (BSP) is defined as follows:

$$(BSP) \quad \underset{\pi \in S}{\text{Min}} \left\{ \begin{array}{ll} \dfrac{1}{\sum_{j=1}^{n} w^1_{\pi(j)}} & \sum_{j=1}^{n} w^1_{\pi(j)} F_{\pi(j)} \\ \dfrac{1}{\sum_{j=1}^{n} w^2_{\pi(j)}} & \sum_{j=1}^{n} w^2_{\pi(j)} F_{\pi(j)} \end{array} \right\}$$

2.3 Heuristic Method:

In this section, we describe a method to enumerate a substantial number of efficient solutions of the BSP. The method is heuristic in nature in the sense that it does not generate all the efficient solutions. The motivation behind this method is to generate a reasonable number of efficient solutions which sufficiently represent the whole set of efficient solutions and which is easy to understand and efficient to implement. This method is essentially based on the trade-off between the two objective functions.

The algorithm first of all determines a sequence, using WSPT rule, which is optimum with respect to the first objective function. Then by suitable adjacent pairwise exchanges, it decreases the value of second objective function without increasing the value of the first objective function. This yields the first efficient solution. Then, by similar adjacent pairwise exchanges, value of the second objective function is decreased by incurring minimum increase in the value of first objective function. In this manner, all the efficient solutions are enumerated. The method terminates when value of the second objective function cannot be decreased anymore. The stepwise description of this method is given below:

Step 1: Set k = 1, solve the single criterion scheduling problem with weights as $w^1_{\pi(j)}$. Let π^1 be the optimum sequence. Go to Step 2.

Step 2: Identify a pair of adjacent jobs i, j in π^1 such that $r^1_i = r^1_j$ and $r^2_i < r^2_j$ where $r^1_i = w^1_i/p_i$ and $r^2_i = w^2_i/p_i$. If no such pair exists, go to Step 3, otherwise interchange i and j, update π^1 and repeat this step.

Step 3: Let A = {(i, j)/job i and job j are adjacent in π^k and $r^2_i < r^2_j$ }. If A is empty, go to Step 4, otherwise define the ratio R (i, j) for each (i, j) in A as follows:

$$R(i,j) = \{(r_j^2 - r_i^2)/(r_i^1 - r_j^1)\}$$

Select the pair $(k, \ell) \in A$ such that

$$R(k, \ell) = \underset{(i,j) \in A}{\text{Max.}} \{ R(i, j)\},$$

and interchange the jobs k and ℓ. Let π^{k+1} be the sequence thus obtained. Set k = k+1 and repeat this step.

Step 4: $(\pi^1, \pi^2, \ldots, \pi^k)$ are the efficient solutions of the BSP.

Complexity of the Algorithm:

Theorem 2.1:

The algorithm enumerates atmost $\frac{n(n-1)}{2}$ efficient solutions.

Proof:

Let us sequence the jobs in the order of non-decreasing values of r_j^2. We assume without any loss of generality that in this sequence $\pi(j) = j$, for $j = 1, \ldots, n$. It is clear from Step 3 of the algorithm that for a job j to generate another efficient solution by an adjacent pairwise exchange must satisfy the inequality $r_i^2 < r_j^2$. Since jobs are sequenced in the order of non-decreasing r_j^2 values, this inequality cannot be satisfied for more than (j-1) jobs. Hence job j would lead to atmost (j - 1) efficient solutions. When this value is summed for all j, we get the number $\frac{n(n-1)}{2}$ which

which is an upper bound on the number of efficient solutions generated by the algorithm.

It is also clear in the description of the algorithm that $O(n)$ computations are performed at each iteration. Hence the complexity of the algorithm is $O(n^3)$.

2.4 Numerical Example:

In this section, a numerical example is solved to illustrate various steps of the method described in the previous section. The data for the numerical example is given below:

j	p_j	w_j^1	w_j^2	r_j^1	r_j^2
1	8	5	5	0.63	0.63
2	2	1	5	0.50	2.50
3	3	5	2	1.67	0.67
4	7	4	4	0.57	0.57
5	6	4	2	0.67	0.33

Step 1: After solving the single criterion scheduling prob- with respect to w_j^1, we obtain the sequence $\pi^1 = (3\text{-}5\text{-}1\text{-}4\text{-}2)$, with $z_1 = 258$ and $z_2 = 335$.

Step 2: No pair (i, j) exists which satisfies the conditi- So $\pi^1 = (3\text{-}5\text{-}1\text{-}4\text{-}2)$ with $z_1 = 258$ and $z_2 = 335$.

Step 3: $A = \{(5, 1), (4, 7)\}$, $R(5, 1) = 7$, $R(4, 2) = 27$. Job 4 and job 2 are interchanged since $R(4, 2)$ is maximum of all such R. $\pi^2 = (3\text{-}5\text{-}1\text{-}2\text{-}4)$ with $z_1 = 259$ and $z_2 = 308$.

Step 3: $A = \{(5, 1), (1, 2)\}$, $R(5, 1) = 5$, $R(1, 2) = 15$. Job 1 and job 2 are interchanged since $R(1, 2)$ is maximum. $\pi^3 = (3\text{-}5\text{-}2\text{-}1\text{-}4)$, $z_1 = 261$ and $z_2 = 278$.

Step 3: $A = \{(5, 2)\}$, $R(5, 2) = 13$. Job 5 and job 2 are interchanged. $\pi^4 = (3\text{-}2\text{-}5\text{-}1\text{-}4)$ with $z_1 = 263$ and $z_2 = 252$.

Step 3: $A = \{(3, 2), (5, 1)\}$, $R(3, 2) = 1.5714$, $R(5, 1) = 7$. Job 5 and job 1 are interchanged since $R(5, 1)$ is maximum. $\pi^5 = (3\text{-}2\text{-}1\text{-}5\text{-}4)$ with $z_1 = 265$ and $z_2 = 238$.

Step 3: $A = \{(3, 2), (5, 4)\}$, $R(3, 2) = 1.5714$, $R(5, 4) = 2.5$. Job 5 and job 4 are interchanged, since $R(5, 4)$ is maximum. $\pi^6 = (3\text{-}2\text{-}1\text{-}4\text{-}5)$ with $z_1 = 269$ and $z_2 = 228$.

Step 3: $A = \{(3, 2)\}$, $R(3, 2) = 1.5714$. Job 3 and job 2 are interchanged. $\pi^7 = (2\text{-}3\text{-}1\text{-}4\text{-}5)$ with $z_1 = 276$ and $z_2 = 217$.

Step 3: A is empty.

Step 4: The following are the efficient solutions of the BSP,

Sequence	z_1	z_2
3-5-1-4-2	258	335
3-5-1-2-4	259	308
3-5-2-1-4	261	278
3-2-5-1-4	263	252
3-2-1-5-4	265	238
3-2-1-4-5	269	228
2-3-1-4-5	276	217

In Fig. (2.1), efficient solutions are plotted in the bicriterion space.

2.5 Computational Results:

A computer program was written in FORTRAN - 10 for the heuristic method described in Sec. 2.3. The program was coded, debugged and tested on DEC-1090 time-sharing computer system. The program occupies 8n words of the core memory.

The main emphasis was laid on (i) to check the percentage number of efficient solutions enumerated by the heuristic method and (ii) to check the effectiveness of the heuristic method in solving large size problems.

The number of efficient solutions enumerated by the heuristic method is compared with the total number of efficient solutions. An explicit enumeration algorithm was used to find the total number of efficient solutions. Problems of size 5, 6, 7 and 8 were solved for this purpose. In all 50 problems

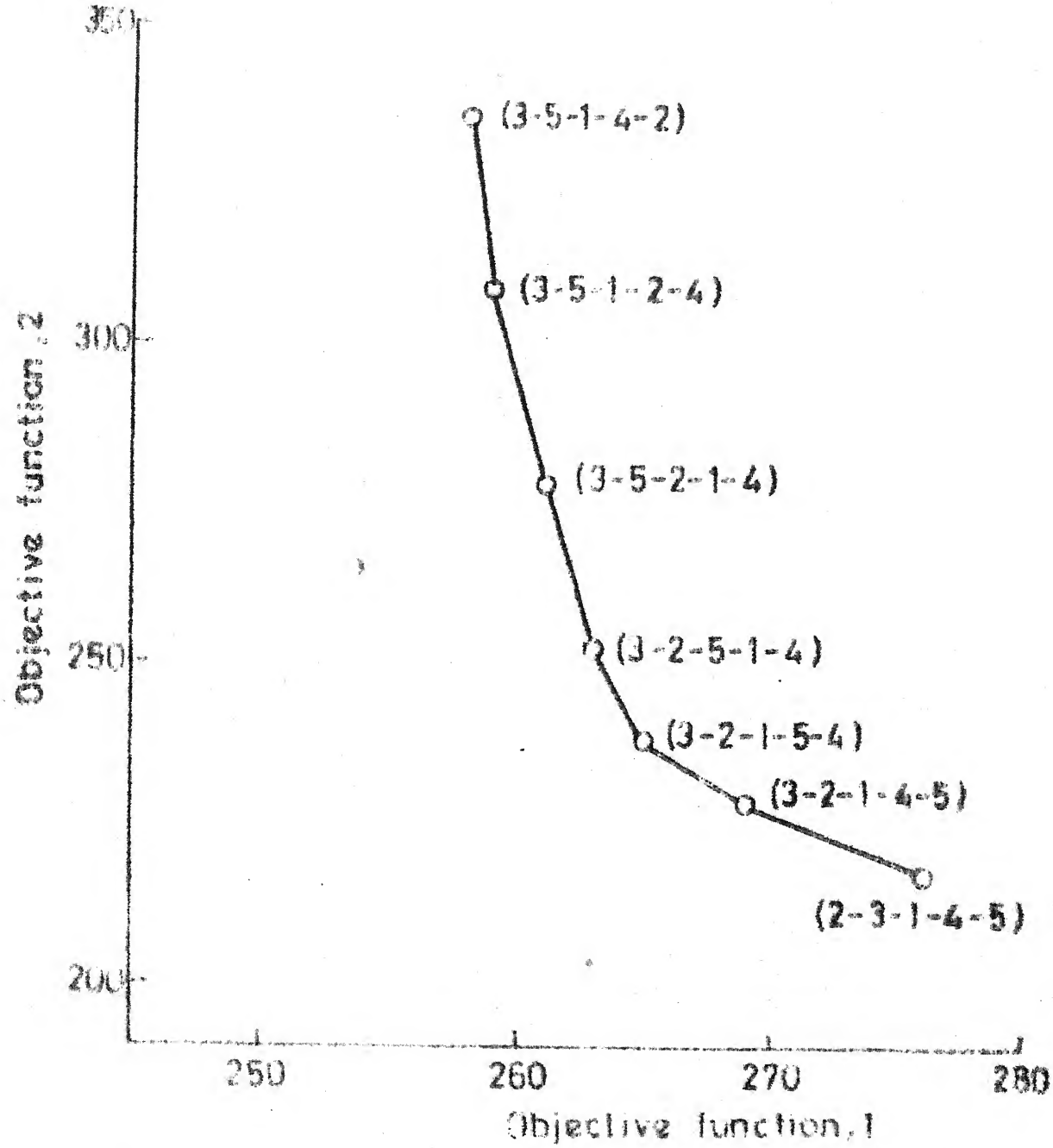

Fig. 2.1 Trade - off curve

of each size were solved. Since it was not possible to solve large sized problems using an explicit enumeration algorithm, this work was limited to the small-sized problems. The number of efficient solutions enumerated by this method were expressed as a percentage of the total number of efficient solutions. Percentage of problems enumerating less than 50 percent, 50 percent to 60 percent and so on were determined. These values are tabulated in Table 2.1. The heuristic method enumerates atleast 50 percent of the total efficient solutions, 90 percent of the time. On an average, it enumerates 100 percent of the efficient solutions 20 percent of the time.

The efficiency of the heuristic method in solving large sized problems is tested by determining the average run-time to solve a problem. Problems of sizes 10, 20, 30, 50, 80 and 100 are solved for this purpose. The results are reported in Table 2.2. The heuristic method is quite efficient in solving large-sized problems. For instance, it solves a 100-job problem in an average run-time of 9 seconds. The average number of efficient solutions enumerated by the heuristic method were also determined for the problems of size specified above. The results are given in Table 2.2. The approximate average run-time to solve an n - job problem can be obtained from the formula $0.122 \times 10^{-4} n^3 - 0.485 \times 10^{-3} n^2 + 0.0167 n - 0.031$, where n is size of the problem,

In Fig. 2.2, a plot is made between size of the problem n, and the average run-time.

In Fig. 2.3, a plot is made between size of the problem n, and the average number of efficient solutions enumerated by the heuristic method.

Table 2.1: Accuracy of the Heuristic Method.

n	Percentage of the efficient solutions enumerated lie between the intervals					
	0 - 50	50 - 60	60 - 70	70 - 80	80 - 90	90-100
5	0	8	6	12	12	62
6	6	12	8	30	14	30
7	5	20	20	20	25	10
8	5	20	25	15	20	15

Table 2.2: Computational Time of the Heuristic Method.

n	Average run-time (in Sec.)	Average Number of Efficient Solutions
10	0.0408	14
20	0.2720	58
30	0.452	128
50	0.9762	363
80	4.565	963
100	8.998	1465

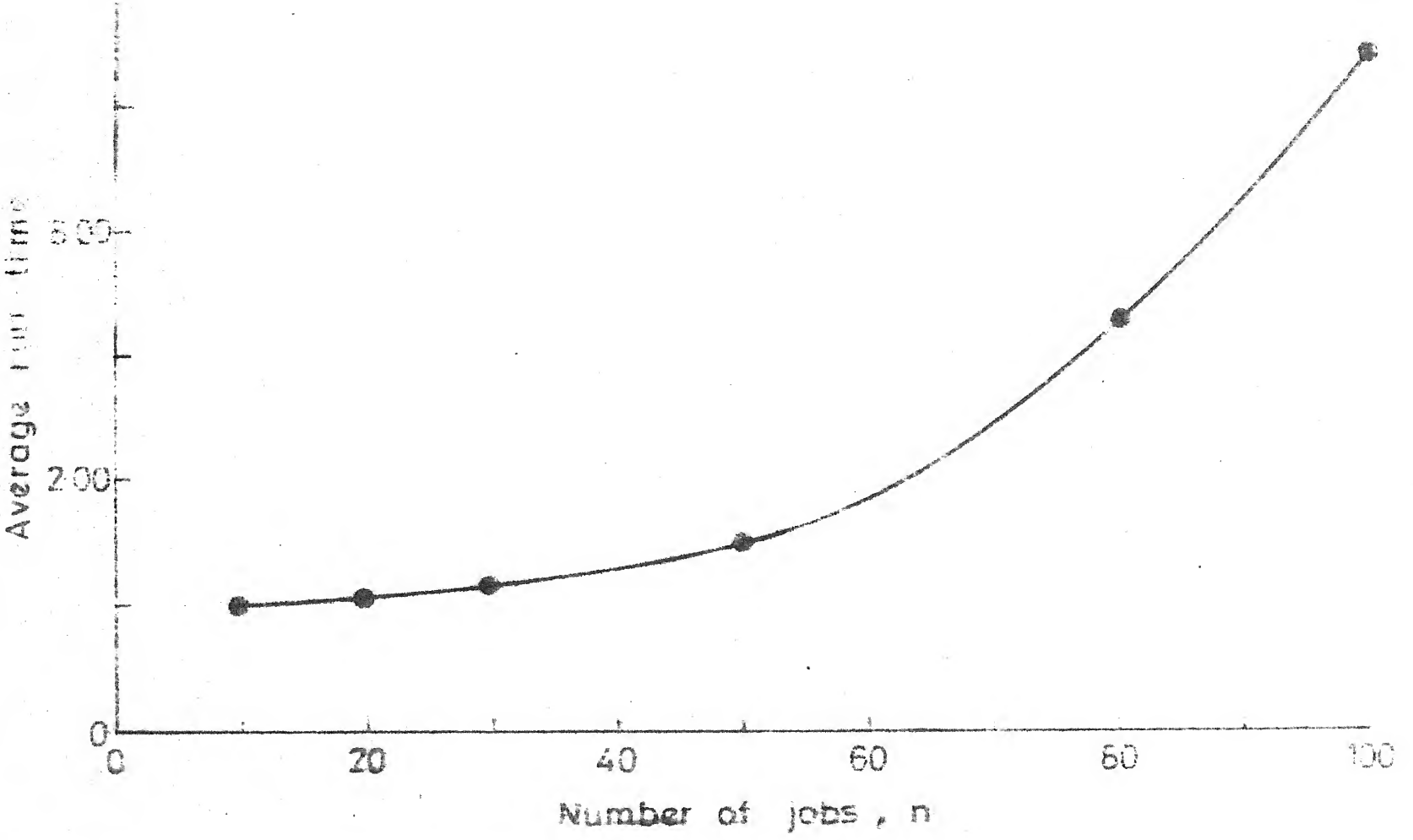

Fig. 2·2 Size of the problem vs average run-time

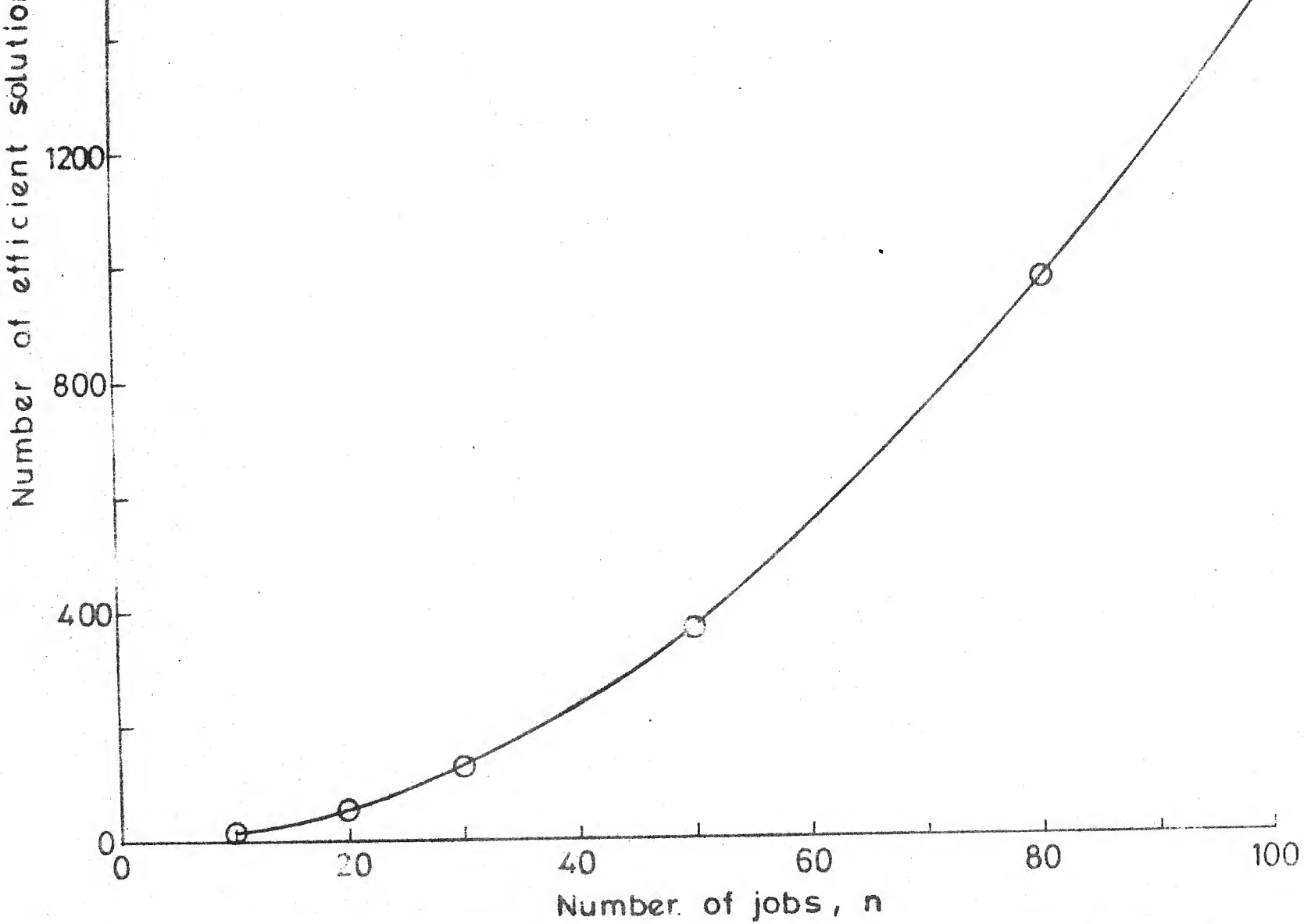

Fig. 2·3 Size of the problem vs average number of efficient solutions

CHAPTER III

CONSTRAINED SCHEDULING PROBLEM-1

3.1 Introduction:

In the previous chapter, we considered multicriteria scheduling problems. These types of problems are important when several criteria must be considered before arriving at the final schedule. Often, the different criteria may not necessarily appear in the objective function. The decision maker may specify certain limits on some criteria and any solution satisfying those limits may be a satisfactory solution. The problem then may be to find the best satisfactory solution with respect to another criteria. Problems of this kind are known as the constrained scheduling problems. In this chapter, we consider one such constrained scheduling problem.

The problem of minimising the sum of weighted completion times subject to a secondary constraint has long been considered to be similar to minimising the sum of completion times subject to the same constraint. Smith [9], presented an algorithm for the n job, one machine scheduling problem where he minimised the sum of weighted completion times subject to the constraint that all the jobs were completed by their due

dates. Recently, Heck and Roberts [6], presented an algorithm for minimising the sum of completion times without increasing the maximum tardiness obtained by the EDD sequence. They claimed that this result could be extended to the problem of weighted completion items in a manner similar to Smith's algorithm. Burns [3], has proved with a counter example that Smith's algorithm did not find the optimum sequence for the problem of minimising the total weighted flowtime subject to the constraint that all the jobs meet their due dates. Hence the claims of Smith [9], and Heck and Roberts [6] regarding the solution of weighted version of the constrained scheduling problem are incorrect.

Recently, Bansal [2] has proposed a branch and bound algorithm to find the optimum solution of the problem of minimising the sum of weighted completion times subject to the constraint that all the jobs meet their due dates. His algorithm uses the breadth first search strategy. He has proposed certain rules to limit the branching. Many of the real life problems are large in size. Bansal's algorithm which is based on branch and bound approach cannot be used for such problems due to high computational requirement. Hence there is a need for heuristic methods which can solve constrained problems of large size within reasonable amount of time and accuracy. The work reported in this chapter is a step in this direction.

In this chapter, a heuristic method is proposed for the n job, one machine sequencing problem to minimise sum of the weighted completion times without increasing the maximum tardiness obtained from the EDD sequence. It is essentially an improvement method which starts with an initial solution and successively improves it until no improvement is possible. Initial solution is obtained by sequencing the jobs in the non-decreasing order of their due dates. In the successive steps, jobs without violating the tardiness criterion are pairwise interchanged. Burns [3] has also given an algorithm which works by pairwise interchanges and he claimed that it yields a local optimum solution. We prove with the help of a counter example that his claim is incorrect.

The heuristic method presented in this chapter was coded in FORTRAN - 10 and extensive computations were performed to test its effectiveness. The results are reported in Sec. 3.8. The computational results are quite encouraging. The heuristic method solves problems of size 50 jobs in an average run time of 15.5 seconds and 90 percent of the time optimal solutions are obtained.

A branch and bound algorithm is also proposed to find the globally optimum solution of the above problem. The proposed algorithm is a modified version of Bansal's algorithm and is computationally more attractive than the latter. The

computational performance of the proposed branch and bound is also reported.

The summary of this chapter is as follows: The problem is stated in Sec. 3.2. In Sec. 3.3, a condition is derived which must be satisfied by a pair of jobs to be interchanged for an improvement in the sum of weighted completion times. In Sec. 3.4, Burns' algorithm is given along with a counter example to show that it does not yield a locally optimum solution. In Sec. 3.5, a heuristic method is proposed which gives a locally optimum solution. In Sec. 3.6, the heuristic method is explained with the aid of a numerical example. In Sec. 3.7, a branch and bound algorithm is presented. Finally computational results are reported in Sec. 3.8.

3.2 The Problem Statement:

Consider the problem of sequencing n jobs on a single machine with 0 ready times. Let $p_{\pi(j)}$ denote the processing time of job in the jth position of sequence π, $c_{\pi(j)}$ be the completion time, $d_{\pi(j)}$ be its due date and $w_{\pi(j)}$ be the weight associated with it. The problem is to minimise the sum of weighted completion times subject to the constraint that maximum tardiness does not exceed the maximum tardiness obtained from the EDD sequence.

The following notations are used,

S : the set of permutation schedules,

π : a permutation schedule,

T : maximum tardiness obtained from EDD sequence.

The problem statement is as follows:

$$\text{Min} \quad \sum_{j=1}^{n} w_{\pi(j)} \, c_{\pi(j)}, \tag{3.1}$$

$$\text{S.t.} \quad c_{\pi(j)} - d_{\pi(j)} \leq T \qquad \text{for } j = 1, \ldots, n \tag{3.2}$$

$$\text{where } c_{\pi(j)} = \sum_{i=1}^{j} p_{\pi(i)}.$$

3.3 Counter Example to Burns' Algorithm:

In this section, we present Burns algorithm for the constrained scheduling problem in Eq. (3.1) - (3.2). Burns has claimed that his algorithm yields a local optimum solution. We present a counter example in this section to show that his algorithm does not yield a local optimum.

Burns' algorithm is as follows:

Step 1: Schedule the jobs in the increasing order of due dates. Let T be the maximum tardiness obtained from the sequence.

Let π be the sequence thus obtained.

Step 2: For k = n-1, n-2, ..., 1 find the first value of k satisfying the following three conditions,

$$T \geq \sum_{i=1}^{n} p_{\pi(i)} - d_{\pi(k)} \tag{3.3}$$

If $p_{\pi(n)} > p_{\pi(k)}$, then

$$c_{\pi(j)} + p_{\pi(n)} - p_{\pi(k)} - d_{\pi(j)} \leq T$$

$$\text{for } j = k+1, \ldots, n-1 \qquad (3.4)$$

$$w_{\pi(k)}\, c_{\pi(k)} + w_{\pi(n)}\, c_{\pi(n)} - w_{\pi(n)}\, c_{\pi(k)}$$

$$- w_{\pi(k)}\, c_{\pi(n)} + (p_{\pi(k)} - p_{\pi(n)}) \sum_{i=k+1}^{n} w_{\pi(i)} > 0 \qquad (3.5)$$

If such k is found, interchange jobs $\pi(k)$ and $\pi(n)$. Reset n to its initial value if necessary and return to the beginning of Step 2. If no k is found satisfying Eqs. (3.3), (3.4) and (3.5) reduce n by 1. For $n \geq 2$, return to the beginning of Step 2. For $n = 1$, the optimum sequence is the current one.

Counter Example:

The data for the counter example is given below:

j	p_j	w_j	d_j
1	7	6	4
2	4	8	4
3	3	3	10
4	4	7	6
5	7	7	2
6	4	4	10
7	4	4	10

Solving the above problem using Burns algorithm, we obtain the sequence 2-4-3-5-1-6-7 with z = 645, where z is the sum of weighted completion times.

Infact, this can be further improved by interchanging jobs 3 and 7 which satisfy the conditions (3.3) - (3.5). The sequence thus obtained in 2-4-7-5-1-6-3 with z = 644. It is obvious from this example that Burns algorithm does not yield the local optimum.

In Sec. 3.5, a heuristic method is proposed which yields a local optimum solution.

3.4 Condition for the Improvement of a Sequence:

In this section, a condition is derived which must be satisfied for a pairwise interchange to result in an improvement in the sum of weighted completion times. Let,

$$\pi = \{1, \ldots, i, \ldots, j, \ldots, n\}$$

and $$\bar{\pi} = \{1, \ldots, j, \ldots, i, \ldots, n\}$$

be two sequences where $\bar{\pi}$ is obtained from π by interchanging the jobs i and j in π. Let c_π be the sum of weighted completion times of the sequence π and $c_{\bar{\pi}}$ be the sum of weighted completion times of the sequence $\bar{\pi}$. Further, let,

$$\Delta c = c_{\bar{\pi}} - c_\pi ,$$

$$M = \sum_{k=1}^{i-1} p_{\pi(k)} = \sum_{k=1}^{i-1} p_{\bar{\pi}(k)} ,$$

$$M' = M + \sum_{k=i+1}^{j-1} p_{\pi}(k) = M + \sum_{k=i+1}^{j-1} p_{\bar{\pi}(k)},$$

$$M_k = M + \sum_{=i+1}^{k} p_{\pi(\ )} = M + \sum_{=i+1}^{k} p_{\bar{\pi}(\ )} \quad \text{for } k > i.$$

$$c = \sum_{k=1}^{i-1} w_{\pi(k)}\, c_{\pi(k)} + w_{\pi(i)}\,(M + p_{\pi(i)})$$

$$+ \sum_{k=i+1}^{j-1} w_{\pi(k)}\,(M_k + p_{\pi(i)}) + w_{\pi(j)}\,(M' + p_{\pi(i)} + p_{\pi(j)})$$

$$+ \sum_{k=j+1}^{n} w_{\pi(k)}\, c_{\pi(k)}$$

$$c_{\bar{\pi}} = \sum_{k=1}^{i-1} w_{\bar{\pi}(k)}\, c_{\bar{\pi}(k)} + w_{\bar{\pi}(i)}\,(M + p_{\bar{\pi}(i)})$$

$$+ \sum_{k=i+1}^{j-1} w_{\bar{\pi}(k)}\,(M_k + p_{\bar{\pi}(i)}) + w_{\bar{\pi}(j)}\,(M + p_{\bar{\pi}(i)} + p_{\bar{\pi}(j)})$$

$$+ \sum_{k=j+1}^{n} w_{\bar{\pi}(k)}\, c_{\bar{\pi}(k)}$$

But $$\sum_{k=1}^{i-1} w_{\pi(k)}\, c_{\pi(k)} = \sum_{k=1}^{i-1} w_{\bar{\pi}(k)}\, c_{\bar{\pi}(k)},$$

$$\sum_{k=j+1}^{n} w_{\pi(k)}\, c_{\pi(k)} = \sum_{k=j+1}^{n} w_{\bar{\pi}(k)}\, c_{\bar{\pi}(k)},$$

$$p_{\bar{\pi}(i)} = p_{\pi(j)}, \quad p_{\bar{\pi}(j)} = p_{\pi(i)},$$

$$w_{\bar{\pi}(i)} = w_{\pi(j)} \quad \text{and} \quad w_{\bar{\pi}(j)} = w_{\pi(i)}.$$

Also, $p_{\pi(k)} = p_{\bar{\pi}(k)}$ and $w_{\pi(k)} = w_{\bar{\pi}(k)}$ for $k = i+1, \ldots, j-1$.

$$\text{So, } c_{\bar{\pi}} = \sum_{k=1}^{i-1} w_{\pi(k)}\, c_{\pi(k)} + w_{\pi(j)}\,(M + p_{\pi(j)})$$

$$+ \sum_{k=i+1}^{j-1} w_{\pi(k)}\,(M_k + p_{\pi(j)}) + w_{\pi(i)}\,(M' + p_{\pi(j)} + p_{\pi(i)})$$

$$+ \sum_{k=j+1}^{n} w_{\pi(k)}\, c_{\pi(k)}\, c_{\pi(k)}.$$

$$\Delta c = c_{\bar{\pi}} - c_{\pi} = w_{\pi(j)}\, M + p_{\pi(j)} \sum_{k=i+1}^{j-1} w_{\pi(k)}$$

$$+ w_{\pi(i)}\, M' + w_{\pi(i)}\, p_{\pi(j)} - w_{\pi(i)}\, M$$

$$- p_{\pi(i)} \sum_{k=i+1}^{j-1} w_{\pi(k)} - w_{\pi(j)}\, M' - w_{\pi(j)}\, p_{\pi(i)}$$

$$= (w_{\pi(i)} - w_{\pi(j)})\,(M' - M) + (p_{\pi(j)} - p_{\pi(i)}) \sum_{k=i+1}^{j-1} w_{\pi(k)}$$

$$+ w_{\pi(i)}\, p_{\pi(j)} - w_{\pi(j)}\, p_{\pi(i)}.$$

$$= (w_{\pi(i)} - w_{\pi(j)}) \sum_{k=i+1}^{j-1} w_{\pi(k)}$$

$$+ (p_{\pi(j)} - p_{\pi(i)}) \sum_{k=i+1}^{j-1} w_{\pi(k)} + (w_{\pi(i)}\, p_{\pi(j)}$$

$$- w_{\pi(j)}\, p_{\pi(i)}). \qquad (3.6)$$

To have an improvement in the sum of weighted completion times from π to $\bar{\pi}$ by interchanging jobs i and j, the

condition $\Delta c < 0$ must be satisfied where Δc is given by Eq. (3.6).

3.5 The Heuristic Method:

In this section, a heuristic method is proposed for the constrained scheduling problem defined by Eqs. (3.1) - (3.2). The initial sequence is obtained by sequencing the jobs in the non-decreasing order of job due dates. In the successive steps, jobs are pairwise interchanged for an improvement in the sum of weighted completion times, without violating the due date constraint. The heuristic method terminates when the objective function cannot be further improved by pairwise exchanges. The heuristic method gives a locally optimum solution. Step by step description of the heuristic is as follows:

Step 1: Let π be the EDD sequence and T be the maximum tardiness of this sequence. Set i = 2, count = 0 and k = 1.

Step 2: Check whether the following conditions are satisfied:

$$T \geq \sum_{j=1}^{i} p_{\pi(j)} - d_{\pi(k)}$$

$$c_{\pi(j)} + p_{\pi(i)} - d_{\pi(j)} \leq T \quad \text{for } j = k+1, \ldots, i-1$$

$$(p_{\pi(i)} - p_{\pi(k)}) \sum_{j=k+1}^{i-1} w_{\pi(j)} + (w_{\pi(k)} - w_{\pi(i)}) \sum_{j=k+1}^{i-1} p_{\pi(j)}$$

$$+ (w_{\pi(k)}\, p_{\pi(i)} - w_{\pi(i)}\, p_{\pi(k)}) < 0$$

If the above conditions are satisfied, then interchange jobs $\pi(i)$ and $\pi(k)$, update the sequence π and increment count by 1. Increment k by 1. If $k \neq i$, go to Step 2, otherwise increment i by 1. If $i \leq n$, set $k = 1$ and go to Step 2. If $i > n$ and count $\neq 0$, set $i = 2$, $k = 1$, count = 0 and go to Step 2, otherwise go to Step 3.

Step 3: π is the optimal sequence.

3.6 Numerical Example:

In this section a numerical example is solved to illustrate various steps of the algorithm. The date of the example is given below:

j	p_j	w_j	d_j
1	4	5	9
2	2	8	6
3	4	3	13
4	1	2	7
5	2	3	11
6	2	4	11
7	1	4	8

Step 1: EDD sequence is π = 2-4-7-1-5-6-3 with z = 204 and T = 3. i = 2 and count = 0.

Step 2: Step 2 is illustrated below.

Iteration 1:

k	i	π(k)	π(i)	Count	z
1	2	-	-	0	204
1	3	-	-	0	204
2	3	4	7	1	202
1	4	-	-	1	202
2	4	-	-	1	202
3	4	-	-	1	202
1	5	-	-	1	202
2	5	-	-	1	202
3	5	-	-	1	202
4	5	-	5	2	200
1	6	-	-	2	200
2	6	-	-	2	200
3	6	-	-	2	200
4	6	-	-	2	200
5	6	1	6	3	194
1	7	-	-	3	194
2	7	-	-	3	194
3	7	-	-	3	194
4	7	-	-	3	194
5	7	-	-	3	194
6	7	-	-	3	194
7	7	-	-	3	194

Iteration 2:

k	i	π(k)	π(i)	Count	z
1	2	-	-	0	194
1	3	-	-	0	194
2	3	-	-	0	194
1	4	-	-	0	194
2	4	-	-	0	194
3	4	-	-	0	194
1	5	-	-	0	194
2	5	-	-	0	194
3	5	4	6	1	193
4	5	5	4	2	192
1	6	-	-	2	192
2	6	-	-	2	192
3	6	-	-	2	192
4	6	-	-	2	192
5	6	-	-	2	192
1	7	-	-	2	192
2	7	-	-	2	192
3	7	-	-	2	192
4	7	-	-	2	192
5	7	-	-	2	192
6	7	-	-	2	192
7	7	-	-	2	192

Iteration 3:

k	i	π(k)	π(i)	Count	z
1	2	-	-	0	192
1	3	-	-	0	192
2	3	-	-	0	192
1	4	-	-	0	192
2	4	-	-	0	192
3	4	-	-	0	192
1	5	-	-	0	192
2	5	-	-	0	192
3	5	-	-	0	192
4	5	-	-	0	192
1	6	-	-	0	192
2	6	-	-	0	192
3	6	-	-	0	192
4	6	-	-	0	192
5	6	-	-	0	192
1	7	-	-	0	192
2	7	-	-	0	192
3	7	-	-	0	192
4	7	-	-	0	192
5	7	-	-	0	192
6	7	-	-	0	192
7	7	-	-	0	192

i = 7, k = 7 and count = 0.

Step 3: The optimal sequence π = 2-7-6-4-5-1-3 and z = 192

3.7 Branch and Bound Algorithm:

In this section, a branch and bound algorithm is proposed to find the global solution of the constrained scheduling problem (3.1) - (3.2). The basic idea in this method is to investigate all feasible solutions by partitioning the solution set into subsets sequentially by a process, known as branching. A lower bound is calculated for the solutions with in each set. If the lower bound is smaller than the objective function value of the incumbent solution and the sequence with the lower bound is feasible, further branching is stopped from this subset. This is known as fathoming. After each partitioning, those subsets with a bound that exceeds that of a known feasible solution are excluded from further partitionings. This is known as pruning. The partitioning continues until a feasible solution is found such that its value is not greater than the bound for any subset. The algorithm terminates when all the subsets are either pruned or fathomed. We now describe various strategies of the branching and bounding algorithm for the constrained scheduling problem considered in this chapter.

A branch and bound algorithm is characterised by its i) bounding strategy, ii) branching strategy, and iii) search strategy. These strategies for the suggested branch and bound algorithm for the constrained scheduling problem (3.1)-(3.2) are as follows (while describing the branch and bound algorithm we use the terms and notations as used by Murty [7]).

Boundary Strategy:

A candidate problem k is completely characterised by two sets A^k and N^k, were A^k is the ordered set of jobs in the partial sequence already assigned, and N^k consists of the jobs yet to be assigned. To determine the lowerbound for the candidate problem k, arrange the jobs in N^k according to the WSPT rule and add them to the end of the partial sequence A^k. The objective function value of this sequence is obviously the lower bound of the candidate problem. If the lower bound value of the candidate problem thus obtained is greater than the objective function value of the incumbent, the candidate problem k is pruned.

However, if the lower bound value is smaller than the objective function value of the incumbent solution, feasibility of the solution is checked by the due date constraint. If the solution is feasible, it is fathomed and it is used to improve the incumbent solution.

Branching and Search Strategy:

If a candidate problem k is neither fathomed nor pruned, it is used for subsequent branching. To perform branching at a candidate problem k, the job with the smallest index, say job in set N^k is selected to be added at the end of the partial sequence A^k satisfying the following due date criterion, the job L must be such that every job in set $N^k - \{L\}$ when put adjacent to job L satisfies the due date constraint.

As a matter of fact, there may be several jobs which can be put after the partial sequence A^k but among the jobs, we select the job with the lowest index. This we do as a matter of convenience and to reduce the storage requirements. However, if no job exists which satisfies the above due date criterion, it is clear that the partial sequence A^k will not lead to a feasible sequence and, therefore, it is dropped from further consideration. Then we move to the parent node of the candidate problem k by remaining the last job in the partial sequence A^k. We perform the branching at this parent node by selecting the unassigned job with smallest index greater than the job presently removed and satisfying the due date constraint.

This search strategy is essentially a modified version of the depth first search strategy and its main advantage is lower computer memory requirement. However, in doing so, it may end up taking more time.

Obtaining the incumbent solution which is used to start the branch and bound algorithm remains yet to be described. As a matter of fact, the near optimum solution obtained by the algorithm described in Sec. 3.4 can be used as the initial incumbent solution. It was verified computationally, that this incumbent solution leads to effective pruning and enables to solve moderately sized constrained scheduling problems optimally.

3.8 Computational Results:

In this section, the computational results obtained from the heuristic method described in Sec. 3.5 and the branch and bound method described in Sec. 3.7 are given. Programs were written in Fortran-10 and tested on the DEC-1090 time-sharing computer system.

The heuristic is computationally tested for its accuracy and its ability in solving large sized problems. To check the accuracy of the algorithm, the solution obtained from the heuristic is compared with that from the branch and bound algorithm. This work is done for the problems of size 7, 9, 10, 12 and 15. It was found that the maximum deviation of the heuristic solution from the global solution was 5 percent. The percentage of problems giving deviation of 0 percent, 1 percent and so on are given in Table 3.1. We observe from the results that on an average, the heuristic yields global solution 90 percent of the times.

To check the efficiency of the algorithm in solving large sized problems, the average run-time taken to solve problems 10, 20, 30, 50, 70 and 80 is determined and reported in Table 3.2. The heuristic is quite effective in solving large sized problems. For instance, the heuristic is able to solve problems of size 50 in 15.72 seconds.

The branch and bound algorithm was also tested for its computational efficiency. The average run-times taken to solve

problems of size 7, 9, 10, 12, 15, 17 and 20 are reported in Table 3.3. The branch and bound algorithm solves problems of size 15 in 15.5 seconds.

Finally, in Fig. 3.1, a plot is made between the size of the problem, n and the average run-time taken by the heuristic to solve the problem. The average run time for a problem of size n is given by the formula,

$$0.839 \times 10^{-6}\, n^4 + 0.265 \times 10^{-3}\, n^3 - 0.0132\, n^2 + 0.229\, n - 1.134,$$

where n is the number of jobs considered.

Table 3.1: Accuracy of the Heuristic.

n	Percentage of problems solved with Δ percent deviation from the global optimum solution.					
	Δ					
	0	1	2	3	4	5
7	96	96	96	97	98	100
9	92	94	94	96	99	100
10	90	92	94	98	98	100
12	91	93	93	95	96	100
15	90	92	94	94	95	100

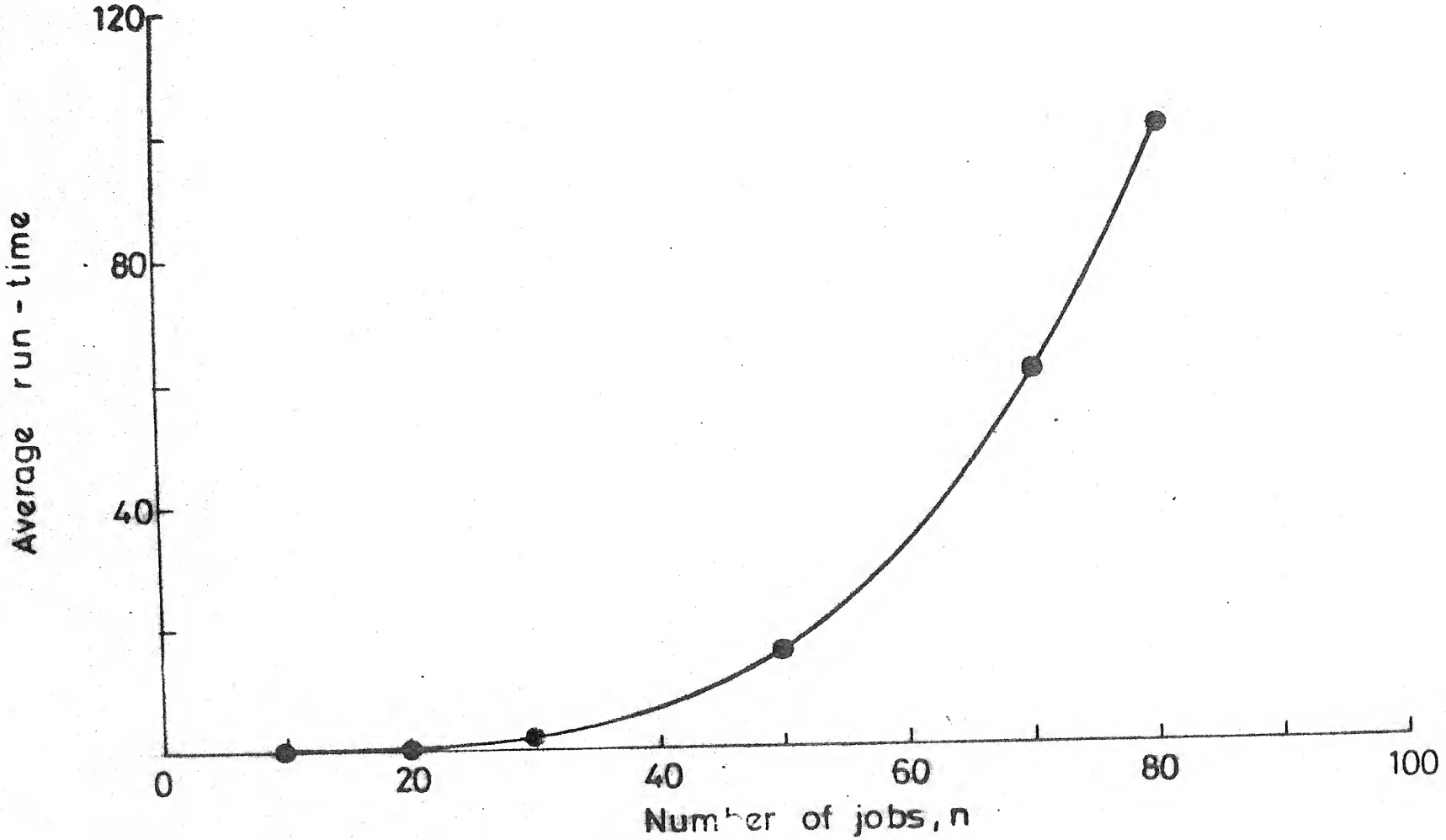

Fig. 3·1 Size of the problem vs average run-time

Table 3.2: Computational time of the Heuristic Method.

n	Average run time (in Sec.)
10	0.130
20	0.360
30	1.788
50	15.725
70	61.504
80	103.075

Table 3.3: Computational Time of the Branch and Bound Algorithm.

n	Average run time (in Sec.)
7	0.98
9	1.52
10	2.45
12	5.01
15	15.54
17	52.17
20	180.02

CHAPTER IV

CONSTRAINED SCHEDULING PROBLEM II

4.1 Introduction:

In Chapter II, a bicriterion scheduling problem was considered. A heuristic method was proposed to enumerate a substantial number of efficient solutions. It is a tedious job to go through the set of efficient solutions (which may be large in number) to pick-up one which is satisfactory. In many practical situations, however, management may desire to improve another performance measure after obtaining an aspiration level of the main measure. Such problems are called the constrained scheduling problems.

In this chapter, the bicriterion scheduling problem considered in Chapter II is changed into a constrained scheduling problem. The first criterion is changed into a constraint with an aspiration level on it. A heuristic method is proposed to solve this problem. The heuristic method yields a locally optimum solution. A branch and bound method is also proposed to test the effectiveness of the heuristic method.

The summary of this chapter is as follows:

In Sec. 4.2, the problem is presented. In Sec. 4.3, a step by step procedure for the proposed heuristic is given. An illustration is presented in Sec. 4.4. Section 4.5 is devoted to the development of a branch and bound procedure for the problem under consideration. Finally, in Sec.4.6, computational results are reported.

4.2 The Problem Statement:

Consider the problem of sequencing n jobs on a single machine with ready times as zero. Let $p_{\pi(j)}$ denote the processing time of job in jth position in π, and $w^1_{\pi(j)}$ and $w^2_{\pi(j)}$ be the two weights associated with it. The problem is to minimise the weighted mean flowtime with respect to w^2_j, subject to the constraint that weighted mean flowtime with respect to w^1_j does not exceed a specified value w_c.

Let, S : be the set of permutation schedules,

π : a permutation schedule, and

$c_{\pi(j)}$: completion time of the job in the jth position in π.

Thus, the problem can be stated as follows:

$$\underset{\pi \in S}{\text{Min}} \quad \frac{1}{W^2} \sum_{j=1}^{n} w^2_{\pi(j)} \, c_{\pi(j)} \tag{4.1}$$

$$\text{S.t.} \quad \frac{1}{W^1} \sum_{j=1}^{n} w^1_{\pi(j)} \, c_{\pi(j)} \leq w_c \,, \tag{4.2}$$

where, $W^2 = \sum_{j=1}^{n} w^2_{\pi(j)}$ and $W^1 = \sum_{j=1}^{n} w^1_{\pi(j)}$.

4.3 The Heuristic Method:

In this section, a heuristic method is proposed to the problem stated in Sec. 4.2. It is an improvement method and is similar to the one presented in the last chapter. The initial solution is obtained by solving the problem as a single criterion scheduling problem with respect to w_j^1. Pairwise interchange of jobs is affected such that an improvement in the second criterion is obtained without violating the constraint (4.2). The method terminates when there are no jobs whose pairwise interchange results in improvement.

The stepwise description of the method is as follows:

Step 1: Let π be the sequence obtained by solving the single criterion problem with respect to weights w_j^1. Set $i = 2$, $k = 1$ and count = 0.

Step 2: Check whether the following conditions are satisfied:

$$\Delta z_2 = (p_{\pi(i)} - p_{\pi(k)}) \sum_{j=k+1}^{i-1} w_{\pi(j)}^2 + w_{\pi(k)}^2 p_{\pi(i)} - w_{\pi(i)}^2 p_{\pi(k)} + (w_{\pi(k)}^2 - w_{\pi(i)}^2) \sum_{j=k+1}^{i-1} p_{\pi(j)} < 0$$

and,

$$z_1 + \Delta z_1 = (p_{\pi(i)} - p_{\pi(k)}) \sum_{j=k+1}^{n} w_{\pi(j)}^1 + w_{\pi(k)}^1 p_{\pi(i)} - w_{\pi(i)}^1 p_{\pi(k)} + (w_{\pi(k)}^1 - w_{\pi(i)}^1) \sum_{j=k+1}^{i-1} p_{\pi(j)} < w_c$$

where z_1, z_2 are the objective function values of criterion 1 and 2 respectively. Δz_1 and Δz_2 are the changes in z_1 and z_2 respectively.

If all the conditions given above are satisfied, interchange $\pi(k)$ and $\pi(i)$, update z_1 and increment count by 1. Increment k by 1. If $k \neq i$, go to Step 2, otherwise increment i by 1. If $i \leq n$, set $k = 1$ and go to Step 2. If $i > n$ and count $\neq 0$, set $i = 2$, count = 0 and $k = 1$ and go to Step 2, otherwise go to Step 3.

Step 3: π is the optimal sequence.

4.4 Numerical Example:

In this section, a numerical example is solved to illustrate various steps of the heuristic method. The data for the example is given in Table 4.1. w_c is 586.

Table 4.1: Data of the Numerical Example.

j	p_j	w_j^1	w_j^2
1	2	8	4
2	8	5	8
3	5	1	2
4	5	6	5
5	5	4	8
6	8	7	5
7	5	4	3

Step 1: Solving the problem as a single criterion problem with respect to w_j^1, we obtain the sequence $\pi = (1\text{-}4\text{-}6\text{-}5\text{-}7\text{-}2\text{-}3)$ with $z_1 = 546$ and $z_2 = 693$. $i = 2$, $k = 1$ and count = 0.

Step 2: Conditions in Step 2 are checked for a pairwise interchange of jobs.

Iteration 1:

k	i	$\pi(k)$	$\pi(i)$	Count	z_1	z_2
1	2	-	-	0	546	693
1	3	-	-	0	546	693
2	3	-	-	0	546	693
1	4	-	-	0	546	693
2	4	4	5	1	572	654
3	4	6	4	2	559	639
1	5	-	-	2	559	639
2	5	-	-	2	559	639
3	5	-	-	2	559	639
4	5	-	-	2	559	639
1	6	-	-	2	559	639
2	6	-	-	2	559	639
3	6	-	-	2	559	639
4	6	6	2	3	585	600
5	6	7	6	4	582	599
1	7	-	-	4	582	599
2	7	-	-	4	582	599
3	7	-	-	4	582	599
4	7	-	-	4	582	599
5	7	-	-	4	582	599
6	7	-	-	4	582	599
7	7	-	-	4	582	599

Since i = 7, k = 7 and count $\neq$ 0, we set i = 2, k = 1 and count = 0 and repeat this step.

Iteration 2:

k	i	$\pi(k)$	$\pi(i)$	Count	z_1	z_2
1	2	-	-	0	582	599
1	3	-	-	0	582	599
2	3	-	-	0	582	599
1	4	-	-	0	582	599
2	4	-	-	0	582	599
3	4	-	-	0	582	599
1	5	-	-	0	582	599
2	5	-	-	0	582	599
3	5	-	-	0	582	599
4	5	-	-	0	582	599
1	6	-	-	0	582	599
2	6	-	-	0	582	599
3	6	-	-	0	582	599
4	6	-	-	0	582	599
5	6	-	-	0	582	599
1	7	-	-	0	582	599
2	7	-	-	0	582	599
3	7	-	-	0	582	599
4	7	-	-	0	582	599
5	7	-	-	0	582	599
6	7	-	-	0	582	599
7	7	-	-	0	582	599

i = 7, k = 7 and count = 0.

Step 3: We obtain the optimal solution as π = (1-5-4-2-6-7-3) with z_1 = 582 and z_2 = 599.

4.5 Branch and Bound Method:

In this section, a branch and bound algorithm is presented to obtain globally optimum solution for the problem given by (4.1) - (4.2). It is quite similar to the branch and bound algorithm presented in the previous chapter. However, the conditions for the feasibility are different.

Bounding Strategy:

The lower bound is calculated with respect to the set of weights w_j^2. Feasibility is checked with respect to constraint (4.2).

Branching and Search Strategy:

The due date constraint is replaced by constraint (4.2). The initial incumbent solution is obtained from the heuristic method.

4.6 Computational Results:

The heuristic as well as the branch and bound algorithm described in previous section were computationally tested. Programs were written in FORTRAN - 10 and debugged and tested on DEC-1090 time sharing computer system.

The heuristic was tested for the accuracy, and efficiency in solving large sized problems. Branch and bound algorithm is used to check the accuracy of the algorithm.

The solution obtained from the heuristic method was compared with that from the branch and bound algorithm for

problems of sizes 6, 7, 8, 9 and 10. The computational results are reported in Table 4.2. On the average, heuristic method yields a globally optimum solution 70 percent of the times. The maximum deviation of the heuristic solution from the global solution was found to be 10 percent in the worst case. The percentage of problems giving a solution with a deviation of 2 percent, 4 percent and so on are given in Table 4.2.

The heuristic was tested on problems of size 10, 20, 30, 50, 80, and 100 to check the efficiency in solving large sized problems. The average run-time to solve a problem of size n, is determined. The results are reported in Table 4.3. The heuristic was able to solve problems with 100 jobs in 7.58 seconds.

Finally, the branch and bound was tested for the efficiency. The average run-time to solve a problem of size, n, was determined. It was not possible to solve a problem of size more than 10 in a reasonable amount of time. The bounds were rather weak in limiting the branching. The results are reported in Table 4.4.

Fig. 4.1 gives a plot which represents the relationship between size of the problems and the average run-time to solve by the heuristic.

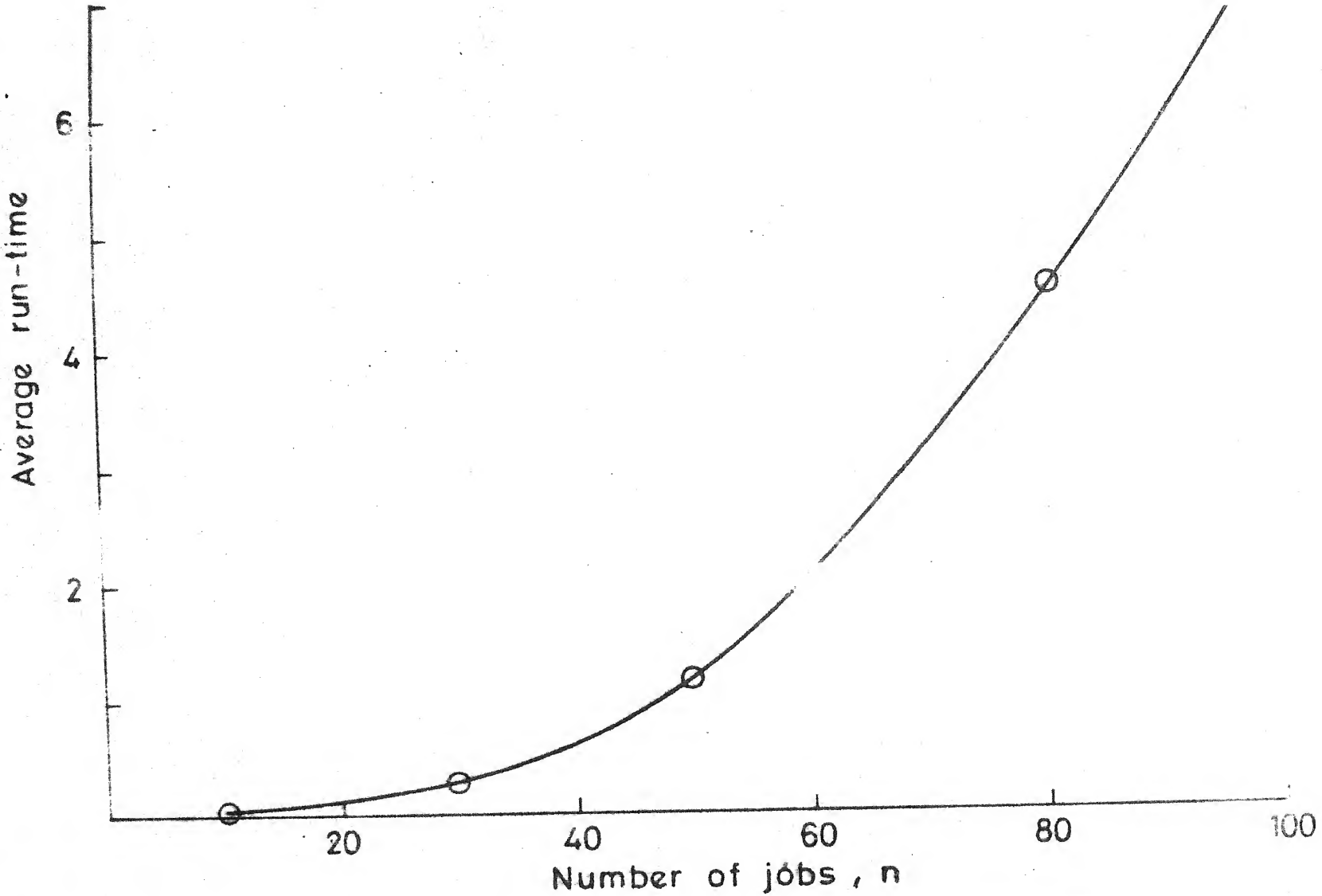

Fig. 4·1 Size of the problem vs average run-tim

Table 4.2: Accuracy of the Heuristic Method.

n	Percentage of the Problems giving the Solution with a Deviation of Δ					
	Δ					
	0	2	4	6	8	10
6	80	88	96	100	100	100
7	76	80	90	96	100	100
8	80	84	84	90	94	100
9	78	80	88	92	96	100
10	70	76	76	88	89	100

Table 4.3: Computational Times of the Heuristic Method.

n	Average Run-Time (in Sec.)
10	0.0153
20	0.088
30	0.260
50	1.090
80	5.155
100	7.530

Table 4.4: Computational Times of the Branch and Bound Algorithm.

n	Average Run-Time (in Sec.)
6	0.320
7	1.632
8	8.720
9	40.210
10	104.340

REFERENCES

1. Baker, K.R., "Introduction to Sequencing and Scheduling," John Wiley and Sons, Inc., New York, 1974.

2. Bansal, S.P., "Single Machine Scheduling to Minimise Weighted Sum of Completion Times with Secondary Criterion - A Branch and Bound Approach," European Journal of Operations Research, 5 (1980), 177-181.

3. Burns, R.N., "Scheduling to Minimise the Weighted Completion Times with Secondary Criterion," Naval Research Logistics Quarterly, 23(1975), 125-129.

4. Emmons, H., "One Machine Sequencing to Minimise Mean Flowtime with Minimum Number Tardy," Naval Research Logistics Quarterly, 22 (1975), 585-592.

5. Emmons, H., "A Note on Scheduling Problem with Dual Criteria," Naval Research Logistics Quarterly, 22(1975), 615-616.

6. Heck, H., and Roberts, S., "A Note on the Extension of a Result on Scheduling with Secondary Criteria," Naval Research Logistics Quarterly, 19 (1972), 403-405.

7. Murty, K.G., "Linear and Combinatorial Programming," John Wiley and Sons, Inc., New York, 1976.

8. Panwalkar, S.S., Dudek, R.A., and Smith, M.L., "Sequencing Research and the Industrial Scheduling Problem," in : S.E. Elmagraby, Ed. Symposium on the Theory of Scheduling Theory and its Applications, Springer, New York, 1973.

9. Smith, W.E., "Various Optimisers for Single Stage Production," Naval Research Logistics Quarterly, 3(1956), 59-66.

10. Vanwessenhove, L., and Gelders, L., "Solving a Bicriterion Scheduling Problem," European Journal of Operations Research, 4(1980), 42-48.